Proyecto fin de grado

Desarrollo de apliciones multiplataforma

Daniel Navacerrada varon

2024

Tabla de contenido

[1. Introducción 2](#_Toc164461264)

[Motivación 2](#_Toc164461265)

[Objetivos 2](#_Toc164461266)

[Planificación 2](#_Toc164461267)

[2. Análisis de requisitos 3](#_Toc164461268)

[Casos de uso 3](#_Toc164461269)

[Tecnologías utilizadas en el desarrollo del proyecto 5](#_Toc164461270)

[Contexto de la base de datos 6](#_Toc164461271)

[Data base Code First 14](#_Toc164461272)

# 1. Introducción

Este proyecto tiene como objetivo desarrollar una aplicación web utilizando el framework Blazor para abordar estas necesidades. La aplicación proporcionará un sistema de gestión de incidencias basado en tickets que podrán llegar a tener un chat entre el usuario de creación con el usuario de soporte que tiene asignado el ticket, permitiendo a los usuarios reportar problemas, asignar responsabilidades y realizar un seguimiento del progreso hasta su resolución. Además, la aplicación incluirá un módulo de control de jornada laboral que registrará los tiempos de entrada y salida de los empleados, facilitando la generación de informes y el cumplimiento de las normativas laborales.

## Motivación

## Objetivos

Los objetivos principales de este proyecto son:

* Diseñar e implementar una aplicación web intuitiva y fácil de usar utilizando Blazor, que permita la gestión eficiente de incidencias y el control de la jornada laboral.
* Investigar y aplicar las mejores prácticas y patrones de diseño en el desarrollo de aplicaciones web con Blazor.
* Demostrar los beneficios de utilizar Blazor para crear aplicaciones web interactivas y receptivas.
* Proporcionar una solución escalable y mantenible que pueda adaptarse a las necesidades cambiantes de las organizaciones.

## Planificación

Para llevar a cabo este proyecto lo que primero tenemos que hacer es definir la line a de tiempo que vamos a llevar para la realización de las distintas fases de la realización de este proyecto, para ello he definido cuatro fases.

Fase 1: Consistirá en la búsqueda de información y formación sobre el entorno de desarrollo con el Framework de Blazor para el desarrollo web en C#.

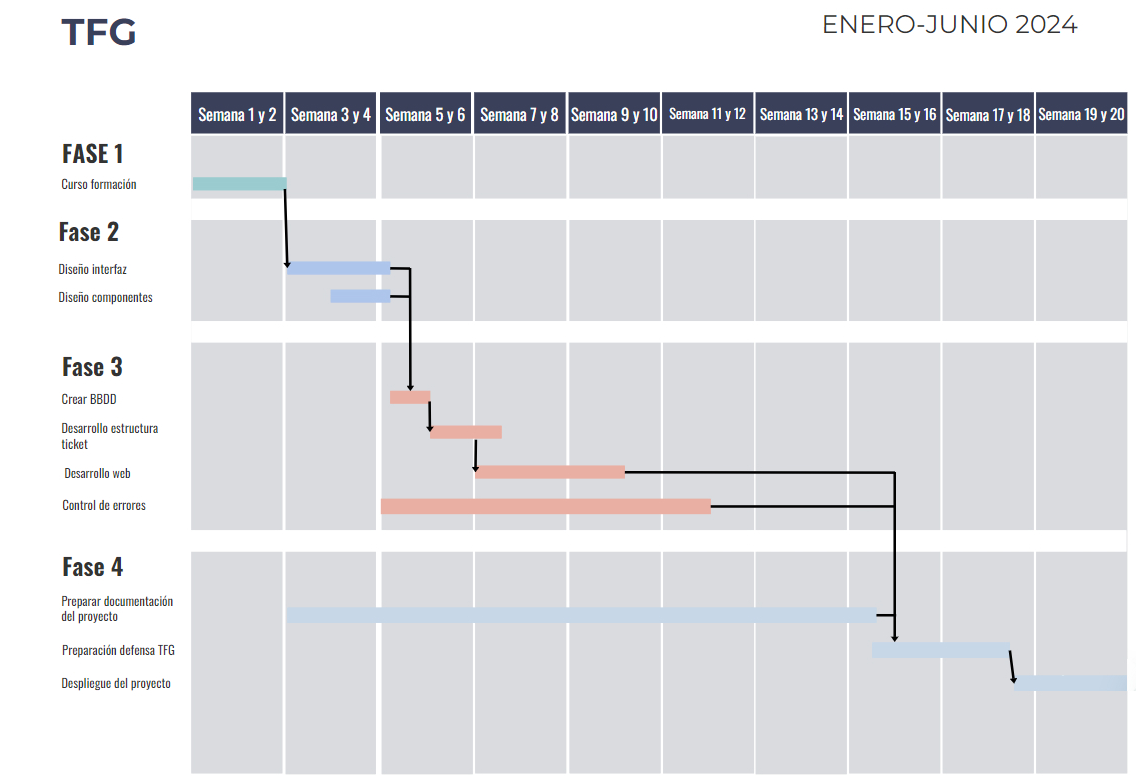
Fase 2: En esta fase se diseñará tanto la interfaz de la web como el diseño lógico y funcional de los distintos componentes/módulos que formarán parte del proyecto.

Fase 3: En esta fase se diseñará el esquema de la base de datos que tendrá el sistema, también se empieza el desarrollo de los componentes y del control de errores de los mismos.

Fase 4: Para esta última fase se terminará de maquetar la documentación del proyecto y se empezará a preparar la defensa y despliegue del mismo para el día de la propia defensa del proyecto y presentación.

A continuación, se añade el diagrama de Gantt que recoge las cuatro fases descritas anteriormente.

Cabe recalcar que estás fases de planificación están sujetas a cambios ya que son la previsión del trabajo que se necesitará para el desarrollo del proyecto.



# 2. Análisis de requisitos

Para este apartado se recogen los requisitos que debe tener el desarrollo del sistema que se va a desarrollar.

## Casos de uso

Con el objetivo de entender el comportamiento del sistema y acciones de los usuarios se realizarán diferentes casos de uso.

A continuación, se distinguirán los actores que se usarán en los casos de uso:

Admin: es el administrador del sistema que tendrá acceso a todo el sistema y podrá manejar todas las tareas de administración de entidades que pertenecen a la web como lo son los roles, usuarios, prioridades y estados de los tickets, consultar las jornadas laborales de los usuarios…

Agente de soporte: Se encargarán de solucionar las incidencias que se registren con los tickets, podrán modificar los estados y prioridades asignados en los tickets, así mismo podrán asignar a otro agente de soporte a un ticket, podrán también crear tickets.

Usuario: sus funciones están muy limitadas ya que solo podrá crear tickets, ver sus propios tickets, iniciar y finalizar su jornada laboral.

Casos de uso:

Entrar a la web

|  |  |
| --- | --- |
| Caso de uso | Entrar a la web |
| Actores | Admin, Agente de soporte, usuario |
| Descripción | Acceso a la web del sistema |
| Condiciones | Iniciar sesión |
| Fujo | Inicia sesión en el sistema.  Realiza las funciones especificas a las que tiene acceso.  Se desconecta del sistema. |

Administración

|  |  |
| --- | --- |
| Caso de uso | Administrar sistema |
| Actor | Admin |
| Descripción | Creación, modificación, eliminación de elementos que influyen en el sistema como los estado y prioridades del sistema, administración de usuarios… |
| Precondiciones | Iniciar sesión teniendo el rol de admin |
| Fujo | Cuando |

## Tecnologías utilizadas en el desarrollo del proyecto

Entity Framework Core 8 de .NET

En este proyecto, se ha utilizado Entity Framework Core en su versión 8 como el mapeador de bases de datos de objeto-relacional (O/RM) principal. Entity Framework Core es una tecnología moderna y eficiente que permite trabajar con bases de datos utilizando objetos .NET, lo que simplifica significativamente el proceso de acceso a datos.

Una de las principales ventajas de utilizar EF Core 8 es que elimina la necesidad de escribir la mayor parte del código de acceso a datos manualmente. En su lugar, se pueden definir modelos de datos utilizando clases de C# y establecer relaciones entre ellos mediante convenciones o configuraciones. EF Core se encarga de generar las consultas SQL necesarias para interactuar con la base de datos, lo que reduce la complejidad y mejora la productividad del desarrollo.

Blazor para el desarrollo web

Para la construcción de la interfaz de usuario web, se ha optado por utilizar Blazor, un innovador framework de desarrollo web creado por Microsoft. Blazor permite crear aplicaciones web interactivas y receptivas utilizando C# en lugar de JavaScript, lo que brinda una experiencia de desarrollo unificada y coherente.

Una de las principales ventajas de Blazor es que permite aprovechar las habilidades existentes de los desarrolladores de .NET, ya que utiliza el mismo lenguaje y ecosistema. Esto facilita la curva de aprendizaje y permite una mayor reutilización de código entre el backend y el frontend de la aplicación.

Componentes de la librería MudBlazor

Para acelerar el desarrollo de la interfaz de usuario y garantizar una apariencia coherente y atractiva, se han utilizado los componentes de la librería MudBlazor. MudBlazor es una librería de componentes de interfaz de usuario para Blazor que sigue las pautas de diseño de Material Design.

Los componentes de MudBlazor son altamente personalizables y proporcionan una amplia gama de elementos de interfaz de usuario prediseñados, como botones, formularios, tablas, diálogos, barras de navegación, entre otros. Estos componentes son fáciles de usar y se integran perfectamente con Blazor, lo que permite un desarrollo rápido y eficiente de la interfaz de usuario.

Además, MudBlazor ofrece un sistema de temas que permite personalizar fácilmente la apariencia de los componentes para adaptarse a la identidad visual del proyecto. Esto garantiza una experiencia de usuario coherente y atractiva en toda la aplicación.

Contexto de la base de datos

El esquema de la base de datos está formado por las siguientes tablas.

**Ticket**: Guarda la información esencial que tendrá el ticket, como el título, la descripción de la incidencia, el estado en el que se encuentra, la prioridad, tanto el usuario que lo ha creado como el agente de soporte que lo lleva, además de las fechas de creación, modificación y finalización.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
| Id | Clave primaria | Integer | False | Identificador único del ticket |
| Title | Atributo | Nvarchar | False | Título de ticket |
| Description | Atributo | Nvarchar | False | Descripción del problema o razón de creación de ticket |
| Status\_Id | Atributo | Integer | False | Estado del ticket |
| Priority\_Id | Atributo | Integer | False | Prioridad del ticket |
| CreationUser | Atributo | Nvarchar | False | Usuario que crea el ticket |
| SupportUser | Atributo | Nvarchar | True | Agente de soporte que resuelve el ticket |
| CreationDate | Atributo | DateTime | True | Fecha de creación del ticket |
| ChangedDate | Atributo | DateTime | True | Fecha en la que el ticket se ha modificado |
| CloseDate | Atributo | DateTime | True | Fecha de cierre del ticket |

**Priority**: Tendrá almacenados las distintas prioridades que tendrán los tickets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de Dato | Nulo | Descripción |
| Id | Clave primaria | Integer | False | Identificador único |
| Nombre | Atributo | Nvarchar | False | Nombre de la prioridad |

**Status**: Se almacenarán los estados que tendrán los tickets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Dato | Tipo de dato | Nulo | Descripción |
| Id | Clave primaria | Integer | False | Identificador único |
| Nombre | Atributo | Nvarchar | False | Nombre del estado |

**Company**: Contendrá los datos básicos de una compañía.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Dato | Tipo de dato | Nulo | Descripción |
| Id | Clave primaria | Integer | False | Identificador único |
| Name | Atributo | Nvarchar | False | Nombre de la compañía |
| Country | Atributo | Nvarchar | True | País de la empresa |
| Address | Atributo | Nvarchar | True | Dirección |
| Cif | Atributo | Nvarchar | False | Código de identificación fiscal de la empresa |
| City | Atributo | Nvarchar | True | Ciudad al la que pertenece la empresa |
| Postal Code | Atributo | Nvarchar | True | Código postal al que pertenece la compañía |

**Attendance**: Guarda los registros de la jornada laboral de los usuarios, teniendo el día y el id del usuario como clave primaria, se encargará de almacenar el día, las fechas completas de entrada y salida y el id del usuario.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
| Date | Clave primaria | Integer | False | Dia en la que se registra la jornada laboral |
| EntryDate | Atributo | DateTime | False | Fecha completa de cuando se inicia la jornada |
| DepartureDate | Atributo | DateTime | True | Fecha completa de cuando se finaliza la jornada laboral |
| UserId | Clave primaria | Nvarchar | False | Identificador único del usuario |
| EntryLongitudCords | Atributo | Nvarchar | False | Coordenadas de longitud de cuando se inicia la jornada |
| EntryLatitudCords | Atributo | Nvarchar | False | Coordenadas de la latitud cuando se inicia la jornada |
| DepartureLongCords | Atributo | Nvarchar | True | Coodenadas de la longitud de cuando se termina la jornada |
| DepartureLattitudCords | Atributo | Nvarchar | True | Coordenadas de latitud de cuando se termina la jornada |

**Chat Message**: Es la tabla en la que se guardaran los mensajes enviados y recibidos de los usuarios, se guardará el emisor y receptor del mensaje el contenido del mensaje y la fecha de emisión del mensaje.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
|  |  |  |  |  |

**AspNetUsers**: Almacena la información básica de los usuarios, como nombre de usuario, correo electrónico, contraseña hasheada, etc. Se puede extender para incluir propiedades adicionales específicas de la aplicación.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
| Id | Clave Primaria | Nvarchar | False | Identificador único |
| Name | Atributo | Nvarchar | True | Nombre del usuario |
| Username | Atributo | Nvarchar | True | Nombre del usuario en el sistema |
| Email | Atributo | Nvarchar | False | Correo electrónico |
| NormalizeEmail | Atributo | Nvarchar | True | Correo electrónico normalizado en mayúsculas |
| NormalizedUsername | Atributo | Nvarchar | True | Nombre en el sistema normalizado en mayúsculas |
| PaswordHash | Atributo | Nvarchar | False | Contraseña cifrada |
| PhoneNumber | Atributo | Nvarchar | True | Número de teléfono |
| EmailConfirmed | Atributo | Bit | False | Confirmación de correo electrónico |
| PhoneNumber Confirmed | Atributo | Bit | False | Confirmación de teléfono |
| TwoFactorEnabled | Atributo | Bit | False | Doble factor de autenticación |
| CompanyId | Clave ajena | Integer | True | Clave ajena de Company |
| LockoutEnd | Atributo | DateTimeOffSet | False | Indica cuando termina el bloqueo de una cuenta |
| LockoutEnable | Atributo | Bit | False | Indica si se puede bloquear al usuario |
| AccessFailedCount | Atributo | Integer | False | Contador de inicios de sesión fallidos |
| SecurityStamp | Atributo | Nvarchar | False | Un valor aleatorio que indica si alguna información de seguridad del usuario ha cambiado. |
| CurrencyStamp | Atributo | Nvarchar | False | Se usa para manejar la concurrencia y asegurar que los datos no se sobrescriban accidentalmente por múltiples solicitudes. |

**AspNetRoles**: Guarda los roles disponibles en la aplicación, lo que permite implementar autorización basada en roles.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
| Id | Clave primaria | Nvarchar | False | Identificador único |
| Name | Atributo | Nvarchar | False | Nombre del Rol |
| NormalizedName | Atributo | Nvarchar | True | Nombre del rol normalizado en mayúsculas |
| ConcurrencyStamp | Atributo | Nvarchar | False | Se usa para manejar la concurrencia y asegurar que los datos no se sobrescriban accidentalmente por múltiples solicitudes. |

**AspNetUserRoles**: Tabla de relación muchos a muchos que vincula los usuarios con los roles que tienen asignados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
| UserId | Clave foránea | Nvarchar | False | hace referencia al Id del usuario en la tabla AspNetUsers. Forma parte de la clave primaria compuesta de esta tabla |
| Rol\_Id | Clave foránea | Nvarchar | False | hace referencia al Id del rol en la tabla AspNetRoles. También forma parte de la clave primaria compuesta. |

**AspNetUserClaims**: Almacena las notificaciones o claims para cada usuario, lo que permite una autenticación más granular basada en claims.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
| Id | Clave primaria | Nvarchar | False | Identificador único |
| UserId | Clave ajena | Nvarchar | False | hace referencia al Id del usuario en la tabla AspNetUsers. Indica a qué usuario pertenece este claim |
| ClaimType | Atributo | Nvarchar | False | Representa el tipo de claim |
| ClaimValue | Atributo | Nvarchar | False | Representa el valor asociado al claim |

**AspNetUserLogins**: Registra los inicios de sesión externos de cada usuario, permitiendo el uso de proveedores de autenticación de terceros como Google, Facebook o Microsoft.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
| LoginProvider | Clave primaria | Nvarchar | False | proveedor de autenticación externo |
| ProvaiderKey | Atributo | Nvarchar | False | representa la clave única del usuario para el proveedor de autenticación externo especificado. |
| ProvaiderDisplayName | Atributo | Nvarchar | False | Nombre del proveedor de autenticación |
| UserId | Clave ajena | Nvarchar | False | hace referencia al Id del usuario en la tabla AspNetUsers. Indica a qué usuario pertenece este login externo |

**AspNetUserTokens**: Guarda los tokens de seguridad generados para cada usuario, como los usados para confirmación de correo electrónico o restablecimiento de contraseña.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Tipo | Tipo de dato | Nulo | Descripción |
| UserId | Clave foranea | Nvarchar | False | hace referencia al Id del usuario en la tabla AspNetUsers |
| LoginProvider | Atributo | Nvarchar | False | indica el proveedor de login para el cual se emitió el token |
| Name | Atributo | Nvarchar | False | nombre del token |
| Value | Atributo | Nvarchar | False | valor del token |

A continuación, se definen las relaciones que tienen entre si las distintas tablas.

ApplicationUser y Company:

Una relación uno a muchos entre ApplicationUser y Company. Cada ApplicationUser está asociado a una Company y una Company tiene asociados a n cantidad de ApplicationUser.

Priority y Ticket

Una relación uno a muchos entre Priority y Ticket. Una Priority puede estar asociada a muchos Tickets, pero un Ticket tiene una única Priority.

Una relación uno a muchos entre Status y Ticket. Un Status puede estar asociado a muchos Tickets, pero un Ticket tiene un único Status.

ApplicationUser (CreationUser), Ticket (CreatedTickets):

Una relación uno a muchos entre ApplicationUser (como CreationUser) y Ticket (como CreatedTickets). Un ApplicationUser puede crear muchos Tickets, pero un Ticket tiene un único CreationUser.

ApplicationUser (SupportUser), Ticket (SupportedTickets):

Una relación uno a muchos entre ApplicationUser (como SupportUser) y Ticket (como SupportedTickets). Un ApplicationUser puede soportar muchos Tickets, pero un Ticket puede tener un único SupportUser (opcional).

ApplicationUser y Attendance:

Una relación uno a muchos entre ApplicationUser y Attendance. Un ApplicationUser puede tener muchas Attendances, pero una Attendance está asociada a un único ApplicationUser.

ApplicationUser (Sender), ChatMessage y ApplicationUser (Receiver), ChatMessage:

Dos relaciones uno a muchos entre ApplicationUser y ChatMessage, una para el Sender y otra para el Receiver. Un ApplicationUser puede enviar y recibir muchos ChatMessages, pero un ChatMessage tiene un único Sender y un único Receiver.

Todo lo descrito anteriormente se podrá visualizar en la imagen adjuntada a continuación.



Data base Code First  
Code First es un enfoque de Entity Framework que permite crear y gestionar la base de datos a partir del código de las clases del modelo de datos. En lugar de diseñar primero la base de datos y luego generar las clases, con Code First se definen las clases y relaciones en el código, y Entity Framework se encarga de crear o actualizar la base de datos automáticamente.

Ventajas de usar Code First en tu proyecto:

* Enfoque centrado en el código: Con Code First, te enfocas principalmente en escribir el código de tus clases de modelo. Esto permite un desarrollo más ágil y centrado en la lógica de negocio, sin preocuparte tanto por los detalles de la base de datos.
* Control sobre el modelo: Tienes un mayor control sobre la definición del modelo de datos. Puedes usar anotaciones o la API fluida para configurar las propiedades, relaciones y restricciones de las entidades.
* Facilita los cambios en el modelo: Si necesitas realizar cambios en el modelo, solo tienes que modificar las clases y luego ejecutar una migración para actualizar la base de datos. Esto simplifica el mantenimiento y la evolución del modelo a lo largo del tiempo.
* Independencia de la base de datos: Code First abstrae los detalles específicos de la base de datos, lo que te permite cambiar fácilmente de proveedor de base de datos sin tener que modificar el código del modelo.
* Migraciones automáticas: Entity Framework incluye un sistema de migraciones que permite aplicar cambios incrementales en la base de datos a medida que evolucionas tu modelo. Esto facilita el control de versiones y la sincronización de la base de datos con el código.
* Pruebas más sencillas: Al tener el modelo definido en código, es más fácil escribir pruebas unitarias y de integración para verificar el comportamiento de las entidades y las operaciones de acceso a datos
* Menor dependencia de la base de datos: Con Code First, puedes trabajar y probar tu aplicación sin necesidad de tener una base de datos física. Puedes usar una base de datos en memoria o una base de datos local para el desarrollo y las pruebas

En resumen, Code First ofrece un enfoque más ágil y centrado en el código para crear y gestionar la base de datos. Proporciona mayor control sobre el modelo, facilita los cambios y las migraciones, y permite una mayor independencia de la base de datos. Es especialmente útil cuando estás desarrollando una nueva aplicación desde cero o cuando quieres tener un mayor control sobre la estructura de la base de datos.